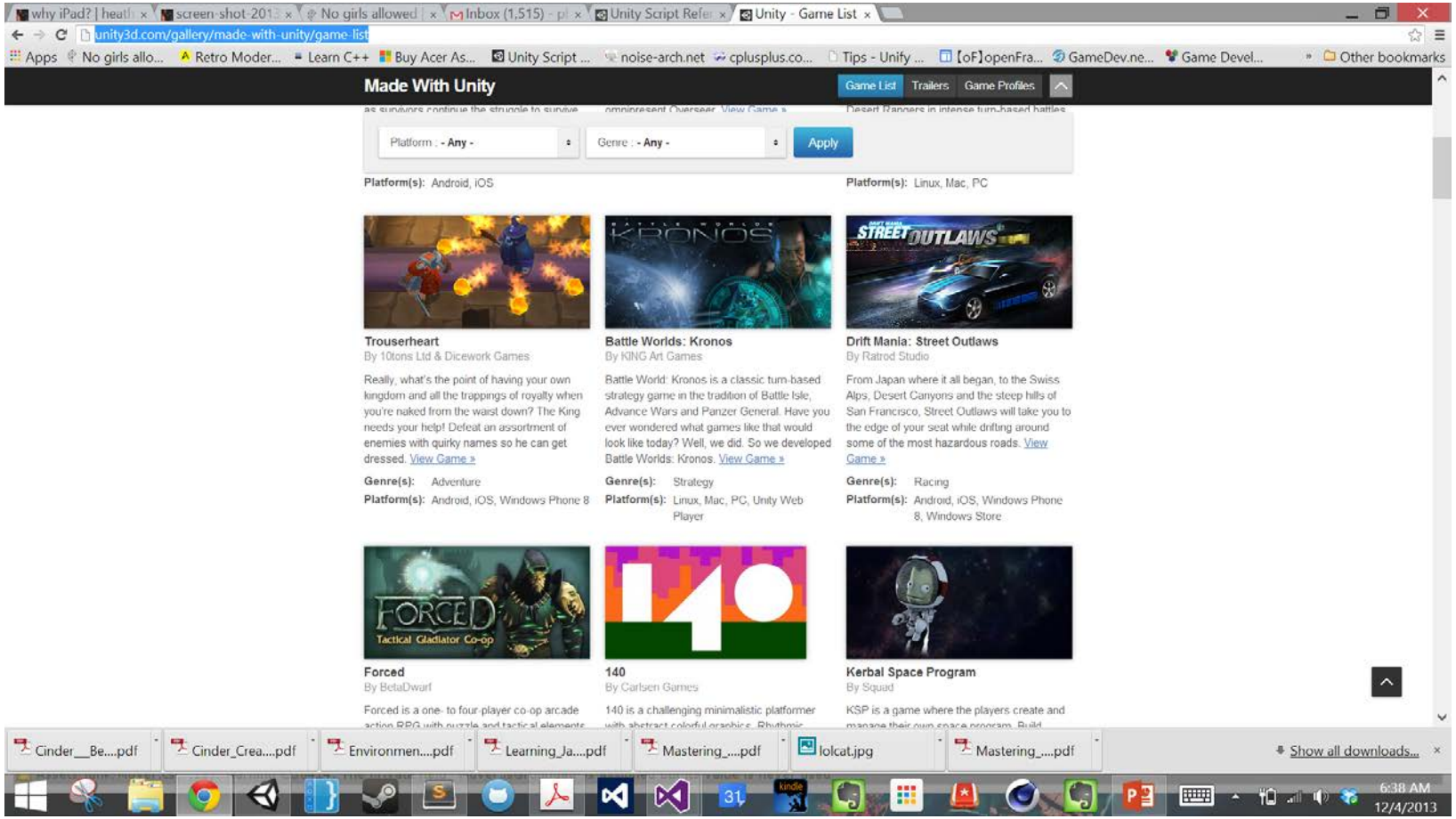
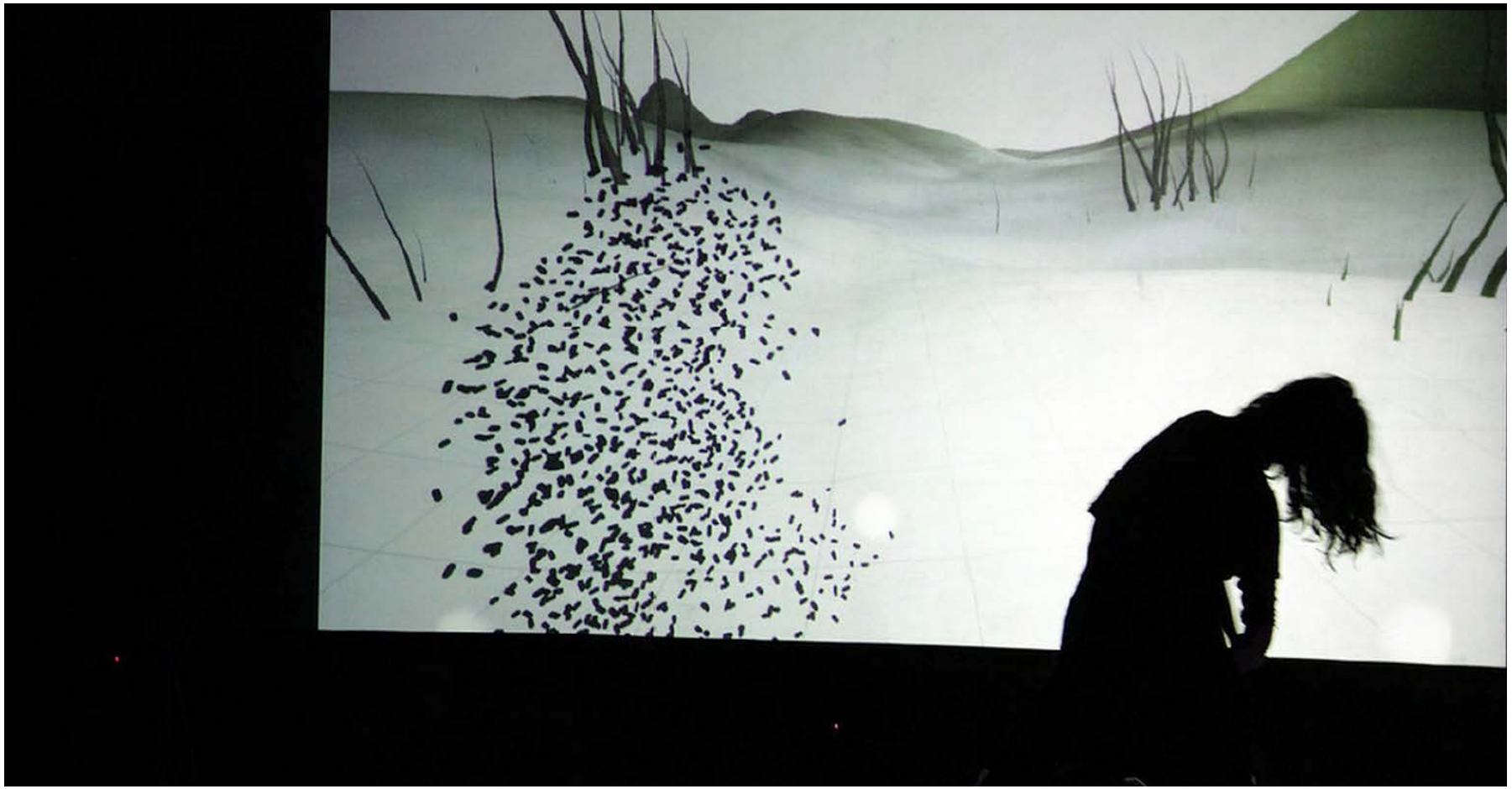
Unity is a Game Engine.



Unity comes with prebuilt functionality speeding development drastically

- Collision detection
- Key presses / input
- Animation engine
- Physics
- Lighting
- Cameras
- Terrain
- 2D + 3D worlds
- Publishing to multiple platforms
- OpenGL set up

Unity is also used for installation



Swimming with Particles

http://www.creativeapplications.net/other/dancing-with-swarming-particles-kinect-unity/

Good indie Unity games

- Tale of Tales
- 140 http://vimeo.com/59001919
- Mirror Moon http://www.youtube.com/watch?v=s2l3h4AeDXI
- Drei
- Panoramical
- Proteus http://www.youtube.com/watch?v=rpkpuoq6y9s

The Unity Platform SDK

Best learning resources:

- http://unity3d.com/learn/documentation
- catlikecoding.com
- http://unitygems.com/
- Unity Forum - http://forum.unity3d.com/forum.php

Everything in Unity is a Game Object.

Think of game objects like empty tool boxes.

- They just are empty boxes waiting to be filled with tools, or COMPONENTS.
- Components do things. They are the tools in the box.
- All game objects have a Transform component that controls position, scale and rotation. This is like your hammer – every tool box has to have one.
- You can script all public properties of any component, allowing you to change things over time, such as animation, color, visibility, user input, on screen text and more

There are many types of tools and not all tool boxes are equal even though they share many of the same components.

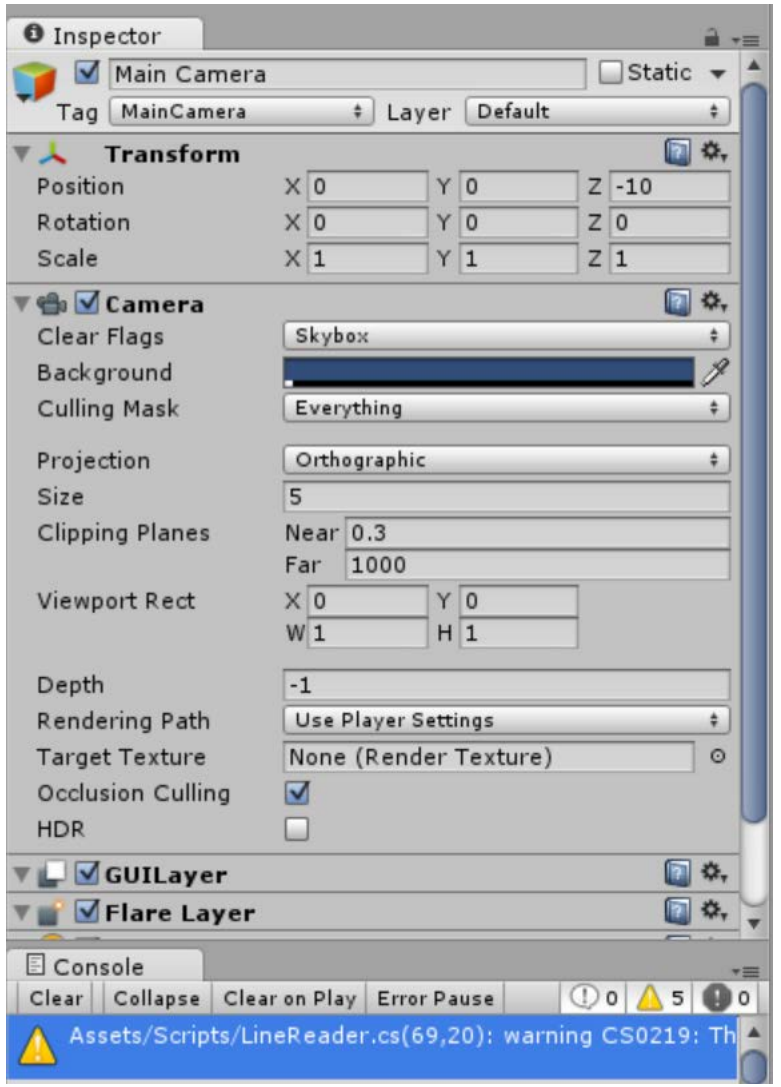
Components do things.

Unity comes with some pre-set up Game Objects with all of the components needed for common tasks

- Particle System, cameras, Cubes, Spheres, Planes, Sprite (2d game object), lights
- All of these pre-setup objects live in the main menu under Game Object.
- There can also be Empty Game Objects in your game. These objects are extremely useful for adding scripts to that create things during the game or use multiple game objects at once.

Unity's Inspector

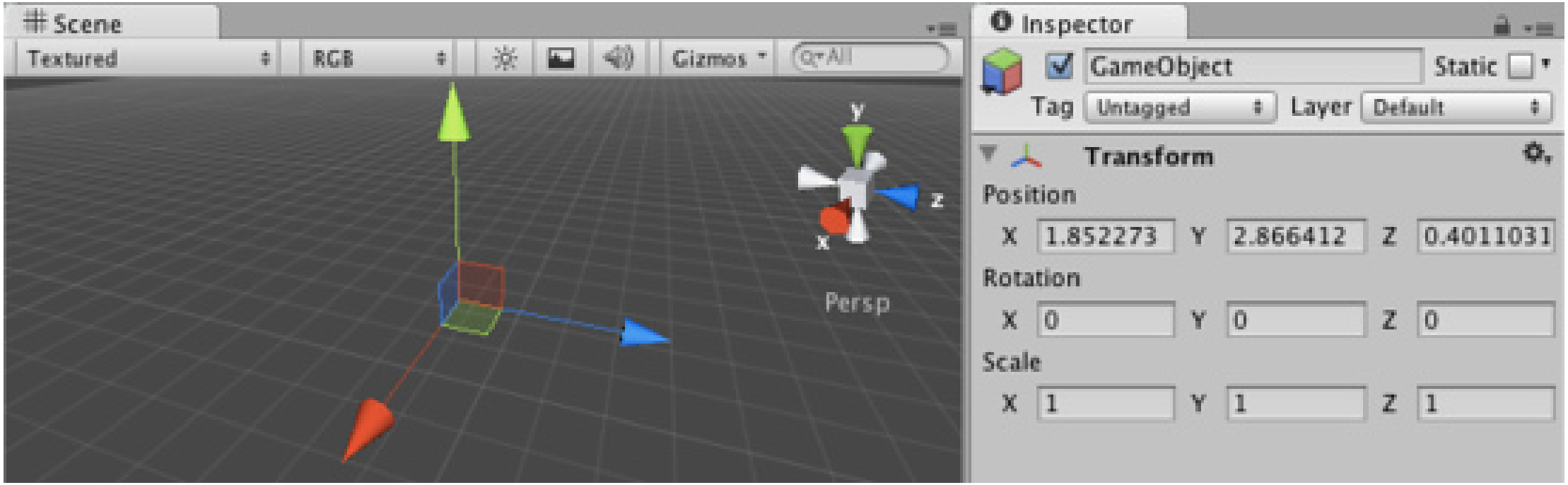
- This area lets you see what components are on a game object and add new components
- Makes visible all component public properties for editing



Other tabs

- Hierarchy – all objects currently in your game world
- Project – your files in your project folder on your computer's hard drive. Not all objects in Project are even in your game. They are just things you might use or will use. It's best to keep this spot organized.
- Game view – What the camera in the game is looking at
- Scene view – A freely positional view that has no connection to the active camera or what the player sees – good for lay out and design.
- Other views exist under Window and are ctrl 0 -9 on the keyboard

The Transform gizmo or x, y, z



Transform tools

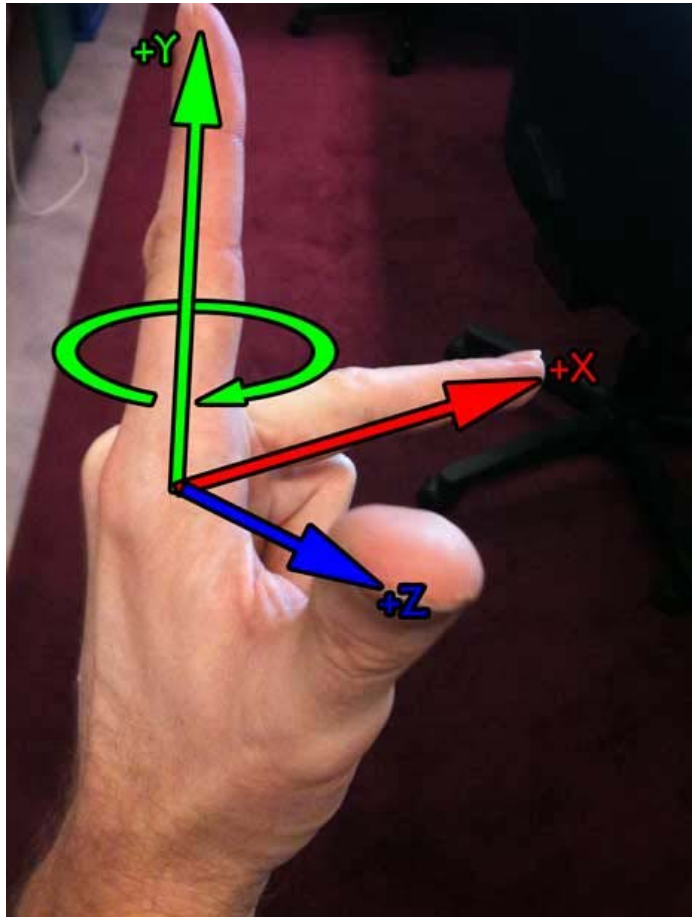
The View, Translate, Rotate, and Scale tools

The tools can be used on any object in the scene. When you click on an object, you will see the tool gizmo appear within it. The appearance of the gizmo depends on which tool is selected.

All three Gizmos can be directly edited in the Scene View.

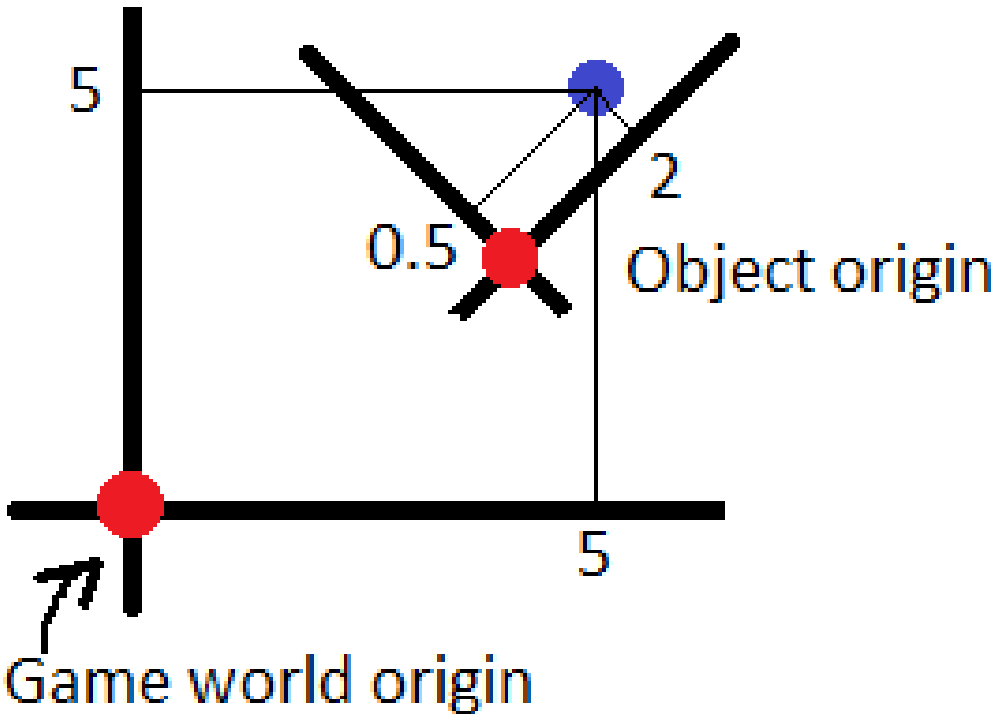
When you click and drag on one of the three gizmo axes, you will notice that its color changes. As you drag the mouse, you will see the object translate, rotate, or scale along the selected axis. When you release the mouse button, the axis remains selected.

Unity is left handed



World space vs Object space

Objects have their own axis and that is different than the world axis, which is always fixed in one direction and. Be warned forward for an object might not be forward for the world.



Rigidbody + Colliders

- Makes the object respond to physics in the game (AKA Gravity and forces). You can also apply torque (rotation)

- Colliders = Track if this object hit something. Colliders can have many shapes, box, circle and polygon. The cheapest are box and circle. Rule of thumb is the more points on the polygon shape, the more processing power it takes for your game to compute the collisions. Colliders are green outlines outside if your object.

Prefab

- Saved game objects. You save these for creation later. You can create these objects (Instantiate) them dynamically with code.

- GameObject b = Instantiate(bullet, position, rotation) as Game Object;

- Nice because you can make it and forget it - it's done. To make a prefab drag an object from the Hierarchy to your Project tab. It will now turn blue in the hierarchy tab and get a cube icon next to it in the Project tab.

Question time!

- What is a rigidbody
- What is a Game Object?
- What is a Component?
- Is Unity left or right handed?
- What is a prefab?
- Bonus points! Key command for rotate?

The final piece of the puzzle. Scripts

- Scripts make components do things over time. Scripts are themselves components
- Scripts have access to anything you can see in the inspector
- Scripts let you LINK game objects together. It’s honestly the magic sauce.

What are types & statements?

- Types are basic kinds of things or objects in your game. For example, a number can be a type.
- Primitive types:
  - Int = a whole number
  - Float = a decimal
  - String = a word
  - Boolean = true or false switch

  Statements are lines of code followed by a semicolon
  int x = 5;
  float y = 9.0f; (in c# you need an f after the last number in a float)

Variables

Variables have a type and they can be changed at any point in time

```
int x = 5;
x=6;
```

Variables can change whenever you need them to! This is really great for things like game score and ship position on screen. These things change over time….

A line of code is called a **Statement**

You can think of a variable like a box

What you put in the box is the type.
An int is a whole number for example.

Functions

- Functions are blocks of instructions –
- They have a few pieces
- Who can use it, what it returns, it's name and parameters passed into it.

```
public void Instructions( ){
        doSomething();
}
private int Instructions ()
{
        int x = 5+6;
        return x;
}
```

Functions are machines.

Function passing

- When you want to run these instructions you call them in your code

```
Instructions();
```

What if you want to put something into your machine? Like water for your coffee. You pass it in in the ( )

```
makeCoffee("water");
```

To pass a variable you need to know it's type

```
public void makeCoffee(String myWater)
{
        Debug.Log("to make coffee you need" + myWater);
}
```

- Primitive variables make copies of themselves when you use them.
- What lives in a function dies in a function.

This is called Scope

- Scopes are defined by these -> { }

```
public void Update{
        //anything make in here is destroyed when you are no longer in here.
        float x = transform.position.x;
}
//this will make an error
Debug.Log(x);
```

You can't get access to x anymore! It's done!

Comments! MAKE THEM!

To make comments just put two slashed in front of your statement

```
// this is a function I wrote to make coffee
MakeCoffee();

/* aldkfjalaflzfj
Asdfjaljadfj
Aksdfjlfjlaf
*/
```

Unity types (objects)

- Unity comes with pre-created types or Objects. These objects are things like
- GameObject, GUIText, RigidBody, RigidBody2D
- When you use one of these objects you use it without creating a copy of it, or you
  “pass it by reference” In fact, these are “reference types”
- You can also define these types yourself! (More on this later.)
- These objects have whole collections of machines and boxes! (functions and variables!)

In fact, you can think of unity game objects like tribbles – they just keep multiplying!

Dot Syntax

- You can get access to those functions and variables using the dot syntax

```
GuiText myText = guiText.text;
Ship.Instructions();
Ship.x = 6;
```

For now just know that these object are made by Untiy and you'll have to learn them from the Sdk. Good news? Unity makes looking them up easy

Scripts are Objects and components! And you define them yourself and attach them to the default Unity Game Objects.

Your Scripts contain all of the functions and variables you might need for your game.

Vectors!

- Points in either 2D 3D space contained one object. A Vector3D

TO create

Vector3 myVect = new Vector3(1.0f, 0.0f, 1.0f);

Vector2 myVect2 = new Vector2(10.2f, 10.1f);

myVect.x = 10.0f;

myVect.z = 1.0f;

Position, rotation and scale in unity is stored as a Vector3 variables

transform.position;

To get access to the x, y, or z positions separately? Use dot syntax

transform.position.x

tranform.rotation.x

Conditionals

- Sets of instructions that run if something is true

Booleans are types which are true or false.

```
bool thisIsTrue = true;


if(thisIsTrue)
{
        run this code
}
```

You can gang them together like you mom would for curfew rules

```
if(thisIsTrue)
{
	//do this;
} else if(thatIsTrue)

{
	//do this;
}
else
{
	//this;
}
```

Unity has 4 main functions

```
void Awake() {
        //if the game object is in the hierarchy, even if the component isn't enabled and visible, this will run.
        // use for initialization and sending messages.
}
void Start () {
        // I  run when an object is put on screen and enabled
}
void Update()
{
        // every frame (variable rate)
}
void Fixed Update ()
{
        //for physics and at a fixed rate
}
```

Public variables let you link objects to scripts

- Any variable set as public is visible in the inspector

```
public int speed = 3;
public GameObject bullet;
```

Goal: Make a space shooter.